R-Skript PUNO-Forschungsprojekt

Teil 1.1 – Objekttypen

Dominik Vogel

Stand: 03.04.2019

Inhaltsverzeichnis

1 Vektoren

Der zweite wichtige Objekttyp wird als Vektor bezeichnet. Ein Vektor enthält mehrere Elemente desselben Datentyps (dazu gleich mehr).

Ein Vektor wird mit `c()` (*combine*) erstellt.

```
alter <- c(21, 78, 24, 26, 35)
alter
```

```
## [1] 21 78 24 26 35
```

Der Vektor `alter` hat nun fünf Elemente, die alle vom Typ `numeric` sind. Mit einem solchen Vektor kann man auch rechnen:

```
alter * 12
```

```
## [1] 252 936 288 312 420
```

`R` multipliziert in diesem Fall alle Werte des Vektors mit 12.

Auch erste Kennzahlen lassen sich mit einem Vektor ermitteln:

```
mean(alter)
```

```
## [1] 36.8
```

Wir können für einen Vektor auch Text verwenden:

```
geschlecht <- c("maennlich", "weiblich")
geschlecht
```

```
## [1] "maennlich" "weiblich"
```

Dieser Vektor hat zwei Elemente vom Typ `character`. Mit einem solchen Vektor kann man natürlich nicht rechnen

```
geschlecht * 2
```

```
## Error in geschlecht * 2: nicht-numerisches Argument für binären Operator
```

Mischt man verschiedene Datentypen, so speichert `R` die Elemente des Vektors wenn möglich in einem Datentyp, der alle Elemente vereinen kann.

```
mix <- c(5, 7, "weiblich")
mix
```

```
## [1] "5"        "7"        "weiblich"
```

An den Anführungszeichen um 5 und 7 sieht man, dass die beiden Werte nicht als Zahlen (`numeric`) sondern als Text (`character`) gespeichert wurden. Damit lässt sich folglich auch nicht mehr rechnen.

```
mix * 5
```

```
## Error in mix * 5: nicht-numerisches Argument für binären Operator
```

2 Matrizen

Matrizen sind im Prinzip Tabellen. Man kann sie auch als Sammlung von Vektoren betrachten. Jede Spalte ist ein Vektor.

Erstellen wir eine Matrix, die in der ersten Spalte das Gewicht mehrerer Personen und in der zweiten Spalte deren Größe speichert. Hierzu erstellen wir zunächst zwei Vektoren und „kleben“ diese anschließend zusammen mit `alter` zu einer Matrix zusammen.

```
gewicht <- c(90, 83, 55, 76, 54)
groesse <- c(191, 150, 165, 170, 171)

X <- cbind(alter, gewicht, groesse)
X
```

```
##      alter gewicht groesse
## [1,]    21      90     191
## [2,]    78      83     150
## [3,]    24      55     165
## [4,]    26      76     170
## [5,]    35      54     171
```

Mit Hilfe von eckigen Klammern kann man auf einzelne Werte zugreifen

```
X[1,3]
```

```
## groesse
##     191
```

Der Befehle gibt den Wert in der ersten Zeile und dritten Spalte aus. Die Grundlegende Logik ist folglich `Matrix[Zeile, Spalte]`.

Man kann auch auf eine gesamte Zeile oder Spalte zugreifen.

```
X[1,]
```

```
##   alter gewicht groesse
##      21      90     191
```

```
X[,3]
```

```
## [1] 191 150 165 170 171
```

3 Data Frame

Der Objekttyp, der uns am meisten beschäftigen wird, ist der Data Frame. Er ist eine Erweiterung von Matrizen und ermöglicht es, Vektoren mit verschiedene Datentypen zu vereinen. Er ist das, was in `SPSS` oder `Stata` der Datensatz ist. Mit dem Unterschied, dass `R` viele verschiedene Datensätze gleichzeitig „geöffnet“ haben kann. Ein Data Frame ist lediglich ein weiteres Objekt.

Erweitern wir unsere Matrix `X` um eine Spalte für Geschlecht:

```
geschlecht <- c("m", "m", "w", "m", "w")
X <- data.frame(alter, gewicht, groesse, geschlecht)
X
```

```
##   alter gewicht groesse geschlecht
## 1    21      90     191          m
## 2    78      83     150          m
## 3    24      55     165          w
## 4    26      76     170          m
## 5    35      54     171          w
```

Data Frames haben nicht nur den Vorteil, dass sie Vektoren unterschiedlicher Art gemeinsam speichern können, sondern auch, dass man die einzelnen Spalten über ihren Namen ansprechen kann:

```
X$groesse
```

```
## [1] 191 150 165 170 171
```

```
X$geschlecht
```

```
## [1] m m w m w
## Levels: m w
```

Für Geschlecht sehen wir nun einen etwas merkwürdigen Output. Der Grund hierfür ist, dass `R` diesen Vektor in einen Faktor umgewandelt hat.

```
class(X$geschlecht)
```

```
## [1] "factor"
```

Faktoren sind Rs Objekttyp zur Speicherung von ordinalen Variablen mit einer begrenzten Anzahl von Ausprägungen. Mehr dazu weiter unten. Wenn wir die Umwandlung in einen Faktor verhindern wollen, können wir in vielen Funktionen die Option `stringsAsFactors` verwenden:

```
X <- data.frame(alter, gewicht, groesse, geschlecht, stringsAsFactors = FALSE)
X
```

```
##   alter gewicht groesse geschlecht
## 1    21      90     191          m
## 2    78      83     150          m
## 3    24      55     165          w
## 4    26      76     170          m
## 5    35      54     171          w
```

`geschlecht` ist nun als normale `character` Variable gespeichert.

```
class(X$geschlecht)
```

```
## [1] "character"
```

4 Faktoren

Faktoren sind Rs Objekttyp zur Speicherung von ordinalen Variablen mit einer begrenzten Anzahl von Ausprägungen. Dieser Datentyp kann an verschiedenen Stellen nützlich sein, weil R die entsprechende Variable automatisch wie eine ordinale Variable behandelt (zum Beispiel bei der Erstellung von Grafiken oder in Regressionen).

Geschlecht hat, wie oben zu sehen, zwei Ausprägungen: m und w.

Es ist sehr einfach, einen Vektor in einen Faktor umzuwandeln:

```
geschlecht_faktor <- factor(geschlecht)
geschlecht_faktor
```

```
## [1] m m w m w
## Levels: m w
```

Man kann die Bezeichnung der Levels auch verändern.

```
levels(geschlecht_faktor)[levels(geschlecht_faktor) == "m"] <- "male"
levels(geschlecht_faktor)[levels(geschlecht_faktor) == "w"] <- "female"
geschlecht_faktor
```

```
## [1] male   male   female male   female
## Levels: male female
```

```
str(geschlecht_faktor)
```

```
##  Factor w/ 2 levels "male","female": 1 1 2 1 2
```

Der Objekttyp Faktor kann manchmal nützlich sein, verursacht aber ungleich häufiger Probleme. Dies ist vor allem dann der Fall, wenn eigentlich numerische Variablen in Faktoren umgewandelt werden und dies Fehlermeldungen verursacht, weil eine bestimmte Funktion ein numerisches Objekt erwartet. Die Importfunktion `read_csv()` verhindert daher auch die Bildung von Faktoren beim Import. Ist eine numerische Variable bereits in einen Faktor umgewandelt worden, hilft die Funktion `as.numeric()`.